

Financial Instrument Pricing Using C++ 2e

DANIEL J. DUFFY

WILEY

Contents

CHAPTER 1

A Tour of C++ and Environs

| | |
|--|----|
| | 1 |
| 1.1 Introduction and Objectives | 1 |
| 1.2 What is C++? | 1 |
| 1.3 C++ as a Multiparadigm Programming Language | 2 |
| 1.4 The Structure and Contents of this Book: Overview | 4 |
| 1.5 A Tour of C++11: Black-Scholes and Environs | 6 |
| 1.5.1 System Architecture | 6 |
| 1.5.2 Detailed Design | 7 |
| 1.5.3 Libraries and Algorithms | 8 |
| 1.5.4 Configuration and Execution | 10 |
| 1.6 Parallel Programming in C++ and Parallel C++ Libraries | 12 |
| 1.7 Writing C++ Applications; Where and How to Start? | 14 |
| 1.8 For Whom is this Book Intended? | 16 |
| 1.9 Next-Generation Design and Design Patterns in C++ | 16 |
| 1.10 Some Useful Guidelines and Developer Folklore | 17 |
| 1.11 About the Author | 18 |
| 1.12 The Source Code and Getting the Source Code | 19 |

CHAPTER 2

New and Improved C++ Fundamentals

| | |
|--|----|
| | 21 |
| 2.1 Introduction and Objectives | 21 |
| 2.2 The C++ Smart Pointers | 21 |
| 2.2.1 An Introduction to Memory Management | 22 |
| 2.3 Using Smart Pointers in Code | 23 |
| 2.3.1 Class <code>std::shared_ptr</code> | 23 |
| 2.3.2 Class <code>std::unique_ptr</code> | 26 |
| 2.3.3 <code>std::weak_ptr</code> | 28 |
| 2.3.4 Should We Use Smart Pointers and When? | 29 |
| 2.4 Extended Examples of Smart Pointers Usage | 30 |
| 2.4.1 Classes with Embedded Pointers | 30 |
| 2.4.2 Re-engineering Object-Oriented Design Patterns | 31 |
| 2.5 Move Semantics and Rvalue References | 34 |
| 2.5.1 A Quick Overview of Value Categories | 34 |
| 2.5.2 Why Some Classes Need Move Semantics | 35 |

| | | |
|--------|---|----|
| 2.5.3 | Move Semantics and Performance | 37 |
| 2.5.4 | Move Semantics and Shared Pointers | 38 |
| 2.6 | Other Bits and Pieces: Usability Enhancements | 39 |
| 2.6.1 | Type Alias and Alias Templates | 39 |
| 2.6.2 | Automatic Type Deduction and the <code>auto</code> Specifier | 41 |
| 2.6.3 | Range-Based <code>for</code> Loops | 42 |
| 2.6.4 | <code>nullptr</code> | 43 |
| 2.6.5 | New Fundamental Data Types | 44 |
| 2.6.6 | Scoped and Strongly Typed Enumerations | 44 |
| 2.6.7 | The Attribute <code>[[deprecated]]</code> | 45 |
| 2.6.8 | Digit Separators | 47 |
| 2.6.9 | Unrestricted Unions | 47 |
| 2.6.10 | <code>std::variant</code> (C++17) and <code>boost::variant</code> | 49 |
| 2.7 | Summary and Conclusions | 52 |
| 2.8 | Exercises and Projects | 52 |

CHAPTER 3

| | | |
|-----------------------------------|--|-----------|
| Modelling Functions in C++ | | 59 |
| 3.1 | Introduction and Objectives | 59 |
| 3.2 | Analysing and Classifying Functions | 60 |
| 3.2.1 | An Introduction to Functional Programming | 60 |
| 3.2.2 | Function Closure | 61 |
| 3.2.3 | Currying | 62 |
| 3.2.4 | Partial Function Application | 62 |
| 3.2.5 | Lambda (Anonymous) Functions | 62 |
| 3.2.6 | Eager and Lazy Evaluation | 63 |
| 3.2.7 | Fold | 63 |
| 3.2.8 | Continuation | 63 |
| 3.3 | New Functionality in C++: <code>std::function<></code> | 64 |
| 3.4 | New Functionality in C++: Lambda Functions and Lambda Expressions | 65 |
| 3.4.1 | Basic Syntax | 65 |
| 3.4.2 | Initial Examples | 66 |
| 3.4.3 | Lambda Functions and Classes: Capturing Member Data | 68 |
| 3.4.4 | Storing Lambda Functions | 69 |
| 3.5 | Callable Objects | 69 |
| 3.6 | Function Adapters and Binders | 70 |
| 3.6.1 | Binding and Function Objects | 72 |
| 3.6.2 | Binding and Free Functions | 73 |
| 3.6.3 | Binding and Subtype Polymorphism | 74 |
| 3.7 | Application Areas | 75 |
| 3.8 | An Example: <i>Strategy</i> Pattern New Style | 75 |
| 3.9 | Migrating from Traditional Object-Oriented Solutions: Numerical Quadrature | 78 |
| 3.10 | Summary and Conclusions | 81 |
| 3.11 | Exercises and Projects | 82 |

CHAPTER 4

| | |
|---|-----------|
| Advanced C++ Template Programming | 89 |
| 4.1 Introduction and Objectives | 89 |
| 4.2 Preliminaries | 91 |
| 4.2.1 Arithmetic Operators and Implicit Conversions | 91 |
| 4.2.2 A Primer on Variadic Functions | 93 |
| 4.2.3 Value Categories | 94 |
| 4.3 <code>decltype</code> Specifier | 94 |
| 4.3.1 Initial Examples | 94 |
| 4.3.2 Extended Examples | 96 |
| 4.3.3 The Auxiliary Trait <code>std::declval</code> | 98 |
| 4.3.4 Expressions, <i>lvalues</i> , <i>rvalues</i> and <i>xvalues</i> | 99 |
| 4.4 Life Before and After <code>decltype</code> | 101 |
| 4.4.1 Extending the STL to Support Heterogeneous Data Types | 103 |
| 4.5 <code>std::result_of</code> and <code>SFINAE</code> | 106 |
| 4.6 <code>std::enable_if</code> | 108 |
| 4.7 <code>Boost enable_if</code> | 112 |
| 4.8 <code>std::decay()</code> Trait | 114 |
| 4.9 A Small Application: Quantities and Units | 115 |
| 4.10 Conclusions and Summary | 118 |
| 4.11 Exercises and Projects | 118 |

CHAPTER 5

| | |
|--|------------|
| Tuples in C++ and their Applications | 123 |
| 5.1 Introduction and Objectives | 123 |
| 5.2 An <code>std::pair</code> Refresher and New Extensions | 123 |
| 5.3 Mathematical and Computer Science Background | 128 |
| 5.4 Tuple Fundamentals and Simple Examples | 130 |
| 5.5 Advanced Tuples | 130 |
| 5.5.1 Tuple Nesting | 130 |
| 5.5.2 Variadic Tuples | 132 |
| 5.6 Using Tuples in Code | 133 |
| 5.6.1 Function Return Types | 133 |
| 5.6.2 Function Input Arguments | 136 |
| 5.7 Other Related Libraries | 138 |
| 5.7.1 Boost Tuple | 138 |
| 5.7.2 Boost Fusion | 139 |
| 5.8 Tuples and Run-Time Efficiency | 140 |
| 5.9 Advantages and Applications of Tuples | 142 |
| 5.10 Summary and Conclusions | 143 |
| 5.11 Exercises and Projects | 143 |

CHAPTER 6

| | |
|--|------------|
| Type Traits, Advanced Lambdas and Multiparadigm Design in C++ | 147 |
| 6.1 Introduction and Objectives | 147 |
| 6.2 Some Building Blocks | 149 |

| | | |
|-------|--|-----|
| 6.3 | C++ Type Traits | 150 |
| 6.3.1 | Primary Type Categories | 151 |
| 6.3.2 | Composite Type Categories | 153 |
| 6.3.3 | Type Properties | 155 |
| 6.3.4 | Type Relationships | 156 |
| 6.3.5 | 'Internal Properties' of Types | 157 |
| 6.3.6 | Other Type Traits | 158 |
| 6.4 | Initial Examples of Type Traits | 158 |
| 6.4.1 | Simple Bridge Pattern | 159 |
| 6.5 | Generic Lambdas | 161 |
| 6.6 | How Useful will Generic Lambda Functions be in the Future? | 164 |
| 6.6.1 | Duck Typing and Avoiding Class Hierarchies | 164 |
| 6.6.2 | Something Completely Different: Homotopy Theory | 167 |
| 6.7 | Generalised Lambda Capture | 171 |
| 6.7.1 | Living Without Generalised Lambda Capture | 173 |
| 6.8 | Application to Stochastic Differential Equations | 174 |
| 6.8.1 | SDE Factories | 176 |
| 6.9 | Emerging Multiparadigm Design Patterns: Summary | 178 |
| 6.10 | Summary and Conclusions | 179 |
| 6.11 | Exercises and Projects | 179 |

CHAPTER 7

Multiparadigm Design in C++

185

| | | |
|-------|---|-----|
| 7.1 | Introduction and Objectives | 185 |
| 7.2 | Modelling and Design | 185 |
| 7.2.1 | Liskov Substitution Principle | 186 |
| 7.2.2 | Single Responsibility Principle | 187 |
| 7.2.3 | An Example: Separation of Concerns for Monte Carlo Simulation | 188 |
| 7.3 | Low-Level C++ Design of Classes | 190 |
| 7.3.1 | Explicit Specifier | 190 |
| 7.3.2 | Deleted and Defaulted Member Functions | 191 |
| 7.3.3 | The <code>constexpr</code> Keyword | 193 |
| 7.3.4 | The <code>override</code> and <code>final</code> Keywords | 195 |
| 7.3.5 | Uniform Initialisation | 197 |
| 7.3.6 | Initialiser Lists | 198 |
| 7.3.7 | Keyword <code>noexcept</code> | 199 |
| 7.4 | Shades of Polymorphism | 199 |
| 7.5 | Is there More to Life than Inheritance? | 206 |
| 7.6 | An Introduction to Object-Oriented Software Metrics | 207 |
| 7.6.1 | Class Size | 207 |
| 7.6.2 | Class Internals | 207 |
| 7.6.3 | Class Coupling | 208 |
| 7.6.4 | Class and Member Function Inheritance | 209 |
| 7.7 | Summary and Conclusions | 210 |
| 7.8 | Exercises and Projects | 210 |

CHAPTER 8

| | |
|--|------------|
| C++ Numerics, IEEE 754 and Boost C++ Multiprecision | 215 |
| 8.1 Introduction and Objectives | 215 |
| 8.1.1 Formats | 216 |
| 8.1.2 Rounding Rules | 217 |
| 8.1.3 Exception Handling | 218 |
| 8.1.4 Extended and Extendible Precision Formats | 219 |
| 8.2 Floating-Point Decomposition Functions in C++ | 219 |
| 8.3 A Tour of <code>std::numeric_limits<T></code> | 221 |
| 8.4 An Introduction to Error Analysis | 223 |
| 8.4.1 Loss of Significance | 224 |
| 8.5 Example: Numerical Quadrature | 224 |
| 8.6 Other Useful Mathematical Functions in C++ | 228 |
| 8.7 Creating C++ Libraries | 231 |
| 8.7.1 Creating Static C++ Libraries | 231 |
| 8.7.2 Dynamic Link Libraries | 237 |
| 8.7.3 Boost C++ DLLs | 239 |
| 8.8 Summary and Conclusions | 239 |
| 8.9 Exercises and Projects | 239 |

CHAPTER 9

| | |
|---|------------|
| An Introduction to Unified Software Design | 245 |
| 9.1 Introduction and Objectives | 245 |
| 9.1.1 Future Predictions and Expectations | 246 |
| 9.2 Background | 247 |
| 9.2.1 Jackson Problem Frames | 248 |
| 9.2.2 The Hatley-Pirbhai Method | 248 |
| 9.2.3 Domain Architectures | 249 |
| 9.2.4 Garlan-Shaw Architecture | 250 |
| 9.2.5 System and Design Patterns | 250 |
| 9.3 System Scoping and Initial Decomposition | 251 |
| 9.3.1 System Context Diagram | 251 |
| 9.3.2 System Responsibilities and Services | 255 |
| 9.3.3 Optimisation: System Context and Domain Architectures | 255 |
| 9.4 Checklist and Looking Back | 259 |
| 9.4.1 A Special Case: Defining the System's Operating Environment | 259 |
| 9.5 Variants of the Software Process: Policy-Based Design | 260 |
| 9.5.1 Advantages and Limitations of PBD | 266 |
| 9.5.2 A Defined Process for PBD | 268 |
| 9.6 Using Policy-Based Design for the DVM Problem | 268 |
| 9.6.1 Introducing Events and Delegates | 272 |
| 9.7 Advantages of Uniform Design Approach | 273 |
| 9.8 Summary and Conclusions | 274 |
| 9.9 Exercises and Projects | 275 |

CHAPTER 10

| | |
|---|------------|
| New Data Types, Containers and Algorithms in C++ and Boost C++ Libraries | 283 |
| 10.1 Introduction and Objectives | 283 |
| 10.2 Overview of New Features | 283 |
| 10.3 C++ <code>std::bitset<N></code> and Boost Dynamic Bitset Library | 284 |
| 10.3.1 Boolean Operations | 286 |
| 10.3.2 Type Conversions | 286 |
| 10.3.3 Boost <code>dynamic_bitset</code> | 287 |
| 10.3.4 Applications of Dynamic Bitsets | 287 |
| 10.4 Chrono Library | 288 |
| 10.4.1 Compile-Time Fractional Arithmetic with <code>std::ratio<></code> | 288 |
| 10.4.2 Duration | 291 |
| 10.4.3 Timepoint and Clocks | 292 |
| 10.4.4 A Simple Stopwatch | 293 |
| 10.4.5 Examples and Applications | 295 |
| 10.4.6 Boost Chrono Library | 300 |
| 10.5 Boost Date and Time | 301 |
| 10.5.1 Overview of Concepts and Functionality | 301 |
| 10.5.2 Gregorian Time | 302 |
| 10.5.3 Date | 302 |
| 10.6 Forwards Lists and Compile-Time Arrays | 306 |
| 10.6.1 <code>std::forward_list<></code> | 306 |
| 10.6.2 <code>boost::array<></code> and <code>std::array<></code> | 309 |
| 10.7 Applications of Boost.Array | 311 |
| 10.8 Boost uBLAS (Matrix Library) | 313 |
| 10.8.1 Introduction and Objectives | 313 |
| 10.8.2 BLAS (Basic Linear Algebra Subprograms) | 313 |
| 10.8.3 BLAS Level 1 | 314 |
| 10.8.4 BLAS Level 2 | 314 |
| 10.8.5 BLAS Level 3 | 315 |
| 10.9 Vectors | 316 |
| 10.9.1 Dense Vectors | 316 |
| 10.9.2 Creating and Accessing Dense Vectors | 317 |
| 10.9.3 Special Dense Vectors | 318 |
| 10.10 Matrices | 318 |
| 10.10.1 Dense Matrices | 319 |
| 10.10.2 Creating and Accessing Dense Matrices | 320 |
| 10.10.3 Special Dense Matrices | 321 |
| 10.11 Applying uBLAS: Solving Linear Systems of Equations | 322 |
| 10.11.1 Conjugate Gradient Method | 323 |
| 10.11.2 LU Decomposition | 325 |
| 10.11.3 Cholesky Decomposition | 327 |
| 10.12 Summary and Conclusions | 330 |
| 10.13 Exercises and Projects | 331 |

CHAPTER 11

| | |
|--|------------|
| Lattice Models Fundamental Data Structures and Algorithms | 333 |
| 11.1 Introduction and Objectives | 333 |
| 11.2 Background and Current Approaches to Lattice Modelling | 334 |
| 11.3 New Requirements and Use Cases | 335 |
| 11.4 A New Design Approach: A Layered Approach | 335 |
| 11.4.1 Layers System Pattern | 338 |
| 11.4.2 Layer 1: Basic Lattice Data Structures | 339 |
| 11.4.3 Layer 2: Operations on Lattices | 342 |
| 11.4.4 Layer 3: Application Configuration | 346 |
| 11.5 Initial '101' Examples of Option Pricing | 347 |
| 11.6 Advantages of Software Layering | 349 |
| 11.6.1 Maintainability | 350 |
| 11.6.2 Functionality | 350 |
| 11.6.3 Efficiency | 351 |
| 11.7 Improving Efficiency and Reliability | 352 |
| 11.8 Merging Lattices | 355 |
| 11.9 Summary and Conclusions | 357 |
| 11.10 Exercises and Projects | 357 |

CHAPTER 12

| | |
|--|------------|
| Lattice Models Applications to Computational Finance | 367 |
| 12.1 Introduction and Objectives | 367 |
| 12.2 Stress Testing the Lattice Data Structures | 368 |
| 12.2.1 Creating Pascal's Triangle | 368 |
| 12.2.2 Binomial Coefficients | 369 |
| 12.2.3 Computing the Powers of Two | 370 |
| 12.2.4 The Fibonacci Sequence | 370 |
| 12.2.5 Triangular Numbers | 371 |
| 12.2.6 Summary: Errors, Defects and Faults in Software | 372 |
| 12.3 Option Pricing Using Bernoulli Paths | 372 |
| 12.4 Binomial Model for Assets with Dividends | 374 |
| 12.4.1 Continuous Dividend Yield | 374 |
| 12.4.2 Binomial Method with a Known Discrete Proportional Dividend | 375 |
| 12.4.3 Perpetual American Options | 376 |
| 12.5 Computing Option Sensitivities | 377 |
| 12.6 (Quick) Numerical Analysis of the Binomial Method | 379 |
| 12.6.1 Non-monotonic (Sawtooth) Convergence | 380 |
| 12.6.2 'Negative' Probabilities and Convection Dominance | 381 |
| 12.6.3 Which Norm to Use when Measuring Error | 381 |
| 12.7 Richardson Extrapolation with Binomial Lattices | 382 |
| 12.8 Two-Dimensional Binomial Method | 382 |
| 12.9 Trinomial Model of the Asset Price | 384 |
| 12.10 Stability and Convergence of the Trinomial Method | 385 |
| 12.11 Explicit Finite Difference Method | 386 |

| | | |
|-------|-------------------------|-----|
| 12.12 | Summary and Conclusions | 389 |
| 12.13 | Exercises and Projects | 389 |

CHAPTER 13

Numerical Linear Algebra: Tridiagonal Systems and Applications **395**

| | | |
|--------|--|-----|
| 13.1 | Introduction and Objectives | 395 |
| 13.2 | Solving Tridiagonal Matrix Systems | 395 |
| 13.2.1 | Double Sweep Method | 396 |
| 13.2.2 | The Thomas Algorithm | 399 |
| 13.2.3 | Examples | 403 |
| 13.2.4 | Performance Issues | 404 |
| 13.2.5 | Applications of Tridiagonal Matrices | 405 |
| 13.2.6 | Some Remarks on Matrices | 405 |
| 13.3 | The Crank-Nicolson and Theta Methods | 406 |
| 13.3.1 | C++ Implementation of the Theta Method for the Heat Equation | 409 |
| 13.4 | The ADE Method for the Impatient | 411 |
| 13.4.1 | C++ Implementation of ADE (Barakat and Clark) for the Heat Equation | 413 |
| 13.5 | Cubic Spline Interpolation | 415 |
| 13.5.1 | Examples | 424 |
| 13.5.2 | Caveat: Cubic Splines with Sparse Input Data | 426 |
| 13.6 | Some Handy Utilities | 427 |
| 13.7 | Summary and Conclusions | 428 |
| 13.8 | Exercises and Projects | 429 |

CHAPTER 14

Data Visualisation in Excel **433**

| | | |
|--------|---|-----|
| 14.1 | Introduction and Objectives | 433 |
| 14.2 | The Structure of Excel-Related Objects | 433 |
| 14.3 | Sanity Check: Is the Excel Infrastructure Up and Running? | 435 |
| 14.4 | ExcelDriver and Matrices | 437 |
| 14.4.1 | Displaying a Matrix | 440 |
| 14.4.2 | Displaying a Matrix with Labels | 441 |
| 14.4.3 | Lookup Tables, Continuous and Discrete Functions | 442 |
| 14.5 | ExcelDriver and Vectors | 444 |
| 14.5.1 | Single and Multiple Curves | 445 |
| 14.6 | Path Generation for Stochastic Differential Equations | 448 |
| 14.6.1 | The Main Classes | 450 |
| 14.6.2 | Testing the Design and Presentation in Excel | 457 |
| 14.7 | Summary and Conclusions | 459 |
| 14.8 | Exercises and Projects | 459 |
| 14.9 | Appendix: COM Architecture Overview | 463 |
| 14.9.1 | COM Interfaces and COM Objects | 465 |
| 14.9.2 | HRESULT and Other Data Types | 466 |
| 14.9.3 | Interface Definition Language | 468 |
| 14.9.4 | Class Identifiers | 468 |

| | | |
|-------|--|-----|
| 14.10 | An Example | 468 |
| 14.11 | Virtual Function Tables | 471 |
| 14.12 | Differences Between COM and Object-Oriented Paradigm | 473 |
| 14.13 | Initialising the COM Library | 474 |

CHAPTER 15

| | | |
|---|---|------------|
| Univariate Statistical Distributions | | 475 |
| 15.1 | Introduction, Goals and Objectives | 475 |
| 15.2 | The Error Function and Its Universality | 475 |
| 15.2.1 | Approximating the Error Function | 476 |
| 15.2.2 | Applications of the Error Function | 477 |
| 15.3 | One-Factor Plain Options | 478 |
| 15.3.1 | Other Scenarios | 485 |
| 15.4 | Option Sensitivities and Surfaces | 488 |
| 15.5 | Automating Data Generation | 491 |
| 15.5.1 | Data Generation Using Random Number Generators: Basics | 492 |
| 15.5.2 | A Generic Class to Generate Random Numbers | 492 |
| 15.5.3 | A Special Case: Sampling Distributions in C++ | 495 |
| 15.5.4 | Generating Numbers Using a Producer-Consumer Metaphor | 497 |
| 15.5.5 | Generating Numbers and Data with STL Algorithms | 498 |
| 15.6 | Introduction to Statistical Distributions and Functions | 499 |
| 15.6.1 | Some Examples | 502 |
| 15.7 | Advanced Distributions | 504 |
| 15.7.1 | Displaying Boost Distributions in Excel | 507 |
| 15.8 | Summary and Conclusions | 511 |
| 15.9 | Exercises and Projects | 511 |

CHAPTER 16

| | | |
|---|--|------------|
| Bivariate Statistical Distributions and Two-Asset Option Pricing | | 515 |
| 16.1 | Introduction and Objectives | 515 |
| 16.2 | Computing Integrals Using PDEs | 516 |
| 16.2.1 | The Finite Difference Method for the Goursat PDE | 517 |
| 16.2.2 | Software Design | 518 |
| 16.2.3 | Richardson Extrapolation | 519 |
| 16.2.4 | Test Cases | 520 |
| 16.3 | The Drezner Algorithm | 521 |
| 16.4 | The Genz Algorithm and the West/Quantlib Implementations | 521 |
| 16.5 | Abramowitz and Stegun Approximation | 525 |
| 16.6 | Performance Testing | 528 |
| 16.7 | Gauss-Legendre Integration | 529 |
| 16.8 | Applications to Two-Asset Pricing | 531 |
| 16.9 | Trivariate Normal Distribution | 536 |
| 16.9.1 | Four-Dimensional Distributions | 542 |
| 16.10 | Chooser Options | 543 |
| 16.11 | Conclusions and Summary | 545 |
| 16.12 | Exercises and Projects | 546 |

CHAPTER 17

| | |
|---|------------|
| STL Algorithms in Detail | 551 |
| 17.1 Introduction and Objectives | 551 |
| 17.2 Binders and <code>std::bind</code> | 554 |
| 17.2.1 The Essentials of <code>std::bind</code> | 554 |
| 17.2.2 Further Examples and Applications | 555 |
| 17.2.3 Deprecated Function Adapters | 556 |
| 17.2.4 Conclusions | 557 |
| 17.3 Non-modifying Algorithms | 557 |
| 17.3.1 Counting the Number of Elements Satisfying a Certain Condition | 558 |
| 17.3.2 Minimum and Maximum Values in a Container | 559 |
| 17.3.3 Searching for Elements and Groups of Elements | 560 |
| 17.3.4 Searching for Subranges | 561 |
| 17.3.5 Advanced Find Algorithms | 563 |
| 17.3.6 Predicates for Ranges | 565 |
| 17.4 Modifying Algorithms | 567 |
| 17.4.1 Copying and Moving Elements | 567 |
| 17.4.2 Transforming and Combining Elements | 569 |
| 17.4.3 Filling and Generating Ranges | 571 |
| 17.4.4 Replacing Elements | 572 |
| 17.4.5 Removing Elements | 573 |
| 17.5 Compile-Time Arrays | 575 |
| 17.6 Summary and Conclusions | 576 |
| 17.7 Exercises and Projects | 576 |
| 17.8 Appendix: Review of STL Containers and Complexity Analysis | 583 |
| 17.8.1 Sequence Containers | 583 |
| 17.8.2 Associative Containers | 583 |
| 17.8.3 Unordered (Associative) Containers | 583 |
| 17.8.4 Special Containers | 584 |
| 17.8.5 Other Data Containers | 584 |
| 17.8.6 Complexity Analysis | 585 |
| 17.8.7 Asymptotic Behaviour of Functions and Asymptotic Order | 585 |
| 17.8.8 Some Examples | 587 |

CHAPTER 18

| | |
|---|------------|
| STL Algorithms Part II | 589 |
| 18.1 Introduction and Objectives | 589 |
| 18.2 Mutating Algorithms | 589 |
| 18.2.1 Reversing the Order of Elements | 590 |
| 18.2.2 Rotating Elements | 590 |
| 18.2.3 Permuting Elements | 592 |
| 18.2.4 Shuffling Elements | 594 |
| 18.2.5 Creating Partitions | 595 |
| 18.3 Numeric Algorithms | 597 |
| 18.3.1 Accumulating the Values in a Container Based on Some Criterion | 597 |
| 18.3.2 Inner Products | 598 |

| | | |
|--------|---|-----|
| 18.3.3 | Partial Sums | 599 |
| 18.3.4 | Adjacent Difference | 600 |
| 18.4 | Sorting Algorithms | 601 |
| 18.4.1 | Full Sort | 601 |
| 18.4.2 | Partial Sort | 602 |
| 18.4.3 | Heap Sort | 603 |
| 18.5 | Sorted-Range Algorithms | 604 |
| 18.5.1 | Binary Search | 604 |
| 18.5.2 | Inclusion | 605 |
| 18.5.3 | First and Last Positions | 606 |
| 18.5.4 | First and Last Possible Positions as a Pair | 607 |
| 18.5.5 | Merging | 608 |
| 18.6 | Auxiliary Iterator Functions | 609 |
| 18.6.1 | <code>advance()</code> | 609 |
| 18.6.2 | <code>next()</code> and <code>prev()</code> | 610 |
| 18.6.3 | <code>distance()</code> | 611 |
| 18.6.4 | <code>iter_swap()</code> | 611 |
| 18.7 | Needle in a Haystack: Finding the Right STL Algorithm | 612 |
| 18.8 | Applications to Computational Finance | 613 |
| 18.9 | Advantages of STL Algorithms | 613 |
| 18.10 | Summary and Conclusions | 614 |
| 18.11 | Exercises and Projects | 614 |

CHAPTER 19

| | |
|--|------------|
| An Introduction to Optimisation and the Solution of Nonlinear Equations | 617 |
| 19.1 Introduction and Objectives | 617 |
| 19.2 Mathematical and Numerical Background | 618 |
| 19.3 Sequential Search Methods | 619 |
| 19.4 Solutions of Nonlinear Equations | 620 |
| 19.5 Fixed-Point Iteration | 622 |
| 19.6 Aitken's Acceleration Process | 623 |
| 19.7 Software Framework | 623 |
| 19.7.1 Using the Mediator to Reduce Coupling | 628 |
| 19.7.2 Examples of Use | 629 |
| 19.8 Implied Volatility | 632 |
| 19.9 Solvers in the Boost C++ Libraries | 632 |
| 19.10 Summary and Conclusions | 633 |
| 19.11 Exercises and Projects | 633 |
| 19.12 Appendix: The Banach Fixed-Point Theorem | 636 |

CHAPTER 20

| | |
|--|------------|
| The Finite Difference Method for PDEs: Mathematical Background | 641 |
| 20.1 Introduction and Objectives | 641 |
| 20.2 General Convection–Diffusion–Reaction Equations and Black–Scholes PDE | 641 |

| | | |
|--------|---|-----|
| 20.3 | PDE Preprocessing | 645 |
| 20.3.1 | Log Transformation | 645 |
| 20.3.2 | Reduction of PDE to Conservative Form | 646 |
| 20.3.3 | Domain Truncation | 647 |
| 20.3.4 | Domain Transformation | 647 |
| 20.4 | Maximum Principles for Parabolic PDEs | 649 |
| 20.5 | The Fichera Theory | 650 |
| 20.5.1 | Example: Boundary Conditions for the One-Factor Black–Scholes PDE | 653 |
| 20.6 | Finite Difference Schemes: Properties and Requirements | 654 |
| 20.7 | Example: A Linear Two-Point Boundary Value Problem | 655 |
| 20.7.1 | The Example | 656 |
| 20.8 | Exponentially Fitted Schemes for Time-Dependent PDEs | 659 |
| 20.8.1 | What Happens When the Volatility Goes to Zero? | 662 |
| 20.9 | Richardson Extrapolation | 663 |
| 20.10 | Summary and Conclusions | 665 |
| 20.11 | Exercises and Projects | 666 |

CHAPTER 21

| | | |
|--------|--|------------|
| | Software Framework for One-Factor Option Models | 669 |
| 21.1 | Introduction and Objectives | 669 |
| 21.2 | A Software Framework: Architecture and Context | 669 |
| 21.3 | Modelling PDEs and Finite Difference Schemes: What is Supported? | 670 |
| 21.4 | Several Versions of Alternating Direction Explicit | 671 |
| 21.4.1 | Spatial Amplification and ADE | 672 |
| 21.5 | A Software Framework: Detailed Design and Implementation | 673 |
| 21.6 | C++ Code for PDE Classes | 674 |
| 21.7 | C++ Code for FDM Classes | 679 |
| 21.7.1 | Classes Based on Subtype Polymorphism | 683 |
| 21.7.2 | Classes Based on CRTP | 685 |
| 21.7.3 | Assembling FD Schemes from Simpler Schemes | 688 |
| 21.8 | Examples and Test Cases | 690 |
| 21.9 | Summary and Conclusions | 693 |
| 21.10 | Exercises and Projects | 694 |

CHAPTER 22

| | | |
|--------|---|------------|
| | Extending the Software Framework | 701 |
| 22.1 | Introduction and Objectives | 701 |
| 22.2 | Spline Interpolation of Option Values | 701 |
| 22.3 | Numerical Differentiation Foundations | 704 |
| 22.3.1 | Mathematical Foundations | 704 |
| 22.3.2 | Using Cubic Splines | 706 |
| 22.3.3 | Initial Examples | 706 |
| 22.3.4 | Divided Differences | 708 |
| 22.3.5 | What is the Optimum Step Size? | 710 |
| 22.4 | Numerical Greeks | 710 |
| 22.4.1 | An Example: Crank–Nicolson Scheme | 712 |

| | | |
|--------|---|-----|
| 22.5 | Constant Elasticity of Variance Model | 715 |
| 22.6 | Using Software Design (GOF) Patterns | 715 |
| 22.6.1 | Underlying Assumptions and Consequences | 717 |
| 22.6.2 | Pattern Classification | 718 |
| 22.6.3 | Patterns: Incremental Improvements | 720 |
| 22.7 | Multiparadigm Design Patterns | 720 |
| 22.8 | Summary and Conclusions | 721 |
| 22.9 | Exercises and Projects | 721 |

CHAPTER 23

| | | |
|--|--|------------|
| A PDE Software Framework in C++11 for a Class of Path-Dependent Options | | 727 |
| 23.1 | Introduction and Objectives | 727 |
| 23.2 | Modelling PDEs and Initial Boundary Value Problems in the Functional Programming Style | 728 |
| 23.2.1 | A Special Case: Asian-Style PDEs | 730 |
| 23.3 | PDE Preprocessing | 731 |
| 23.4 | The Anchoring PDE | 732 |
| 23.5 | ADE for Anchoring PDE | 739 |
| 23.5.1 | The Saul'yev Method and Factory Method Pattern | 744 |
| 23.6 | Useful Utilities | 746 |
| 23.7 | Accuracy and Performance | 748 |
| 23.8 | Summary and Conclusions | 750 |
| 23.9 | Exercises and Projects | 751 |

CHAPTER 24

| | | |
|--|---|------------|
| Ordinary Differential Equations and their Numerical Approximation | | 755 |
| 24.1 | Introduction and Objectives | 755 |
| 24.2 | What is an ODE? | 755 |
| 24.3 | Classifying ODEs | 756 |
| 24.4 | A Palette of Model ODEs | 757 |
| 24.4.1 | The Logistic Function | 757 |
| 24.4.2 | Bernoulli Differential Equation | 758 |
| 24.4.3 | Riccati Differential Equation | 758 |
| 24.4.4 | Population Growth and Decay | 759 |
| 24.5 | Existence and Uniqueness Results | 760 |
| 24.5.1 | A Test Case | 762 |
| 24.6 | Overview of Numerical Methods for ODEs: The Big Picture | 763 |
| 24.6.1 | Mapping Mathematical Functions to C++ | 763 |
| 24.6.2 | Runge–Kutta Methods | 765 |
| 24.6.3 | Richardson Extrapolation Methods | 766 |
| 24.6.4 | Embedded Runge–Kutta Methods | 767 |
| 24.6.5 | Implicit Runge–Kutta Methods | 767 |
| 24.6.6 | Stiff ODEs: An Overview | 768 |
| 24.7 | Creating ODE Solvers in C++ | 770 |
| 24.7.1 | Explicit Euler Method | 771 |
| 24.7.2 | Runge–Kutta Method | 772 |
| 24.7.3 | Stiff Systems | 775 |

| | | |
|-------|-------------------------|-----|
| 24.8 | Summary and Conclusions | 776 |
| 24.9 | Exercises and Projects | 776 |
| 24.10 | Appendix | 778 |

CHAPTER 25

| | | |
|---|--|------------|
| Advanced Ordinary Differential Equations and Method of Lines | | 781 |
| 25.1 | Introduction and Objectives | 781 |
| 25.2 | An Introduction to the Boost <i>Odeint</i> Library | 782 |
| 25.2.1 | Steppers | 782 |
| 25.2.2 | Examples of Steppers | 784 |
| 25.2.3 | Integrate Functions and Observers | 786 |
| 25.2.4 | Modelling ODEs and their Observers | 787 |
| 25.3 | Systems of Stiff and Non-stiff Equations | 791 |
| 25.3.1 | Scalar ODEs | 791 |
| 25.3.2 | Systems of ODEs | 792 |
| 25.4 | Matrix Differential Equations | 796 |
| 25.5 | The Method of Lines: What is it and what are its Advantages? | 799 |
| 25.6 | Initial Foray in Computational Finance: MOL for One-Factor Black-Scholes PDE | 801 |
| 25.7 | Barrier Options | 806 |
| 25.8 | Using Exponential Fitting of Barrier Options | 808 |
| 25.9 | Summary and Conclusions | 808 |
| 25.10 | Exercises and Projects | 809 |

CHAPTER 26

| | | |
|---|---|------------|
| Random Number Generation and Distributions | | 819 |
| 26.1 | Introduction and Objectives | 819 |
| 26.2 | What is a Random Number Generator? | 820 |
| 26.2.1 | Uniform Random Number Generation | 820 |
| 26.2.2 | Polar Marsaglia Method | 820 |
| 26.2.3 | Box-Muller Method | 821 |
| 26.3 | What is a Distribution? | 821 |
| 26.3.1 | Analytical Solutions for Random Variate Computations | 822 |
| 26.3.2 | Other Methods for Computing Random Variates | 823 |
| 26.4 | Some Initial Examples | 825 |
| 26.4.1 | Calculating the Area of a Circle | 826 |
| 26.5 | Engines in Detail | 827 |
| 26.5.1 | Seeding an Engine | 828 |
| 26.5.2 | Seeding a Collection of Random Number Engines | 829 |
| 26.6 | Distributions in C++: The List | 830 |
| 26.7 | Back to the Future: C-Style Pseudo-Random Number Generation | 831 |
| 26.8 | Cryptographic Generators | 833 |
| 26.9 | Matrix Decomposition Methods | 833 |
| 26.9.1 | Cholesky (Square-Root) Decomposition | 835 |
| 26.9.2 | LU Decomposition | 840 |
| 26.9.3 | QR Decomposition | 842 |

| | | |
|---------|---|-----|
| 26.10 | Generating Random Numbers | 845 |
| 26.10.1 | Appendix: Overview of the <i>Eigen</i> Matrix Library | 846 |
| 26.11 | Summary and Conclusions | 848 |
| 26.12 | Exercises and Projects | 849 |

CHAPTER 27

| | | |
|--|---|------------|
| Microsoft .Net, C# and C++11 Interoperability | | 853 |
| 27.1 | Introduction and Objectives | 853 |
| 27.2 | The Big Picture | 854 |
| 27.3 | Types | 858 |
| 27.4 | Memory Management | 859 |
| 27.5 | An Introduction to Native Classes | 861 |
| 27.6 | Interfaces and Abstract Classes | 861 |
| 27.7 | Use Case: C++/CLI as ‘Main Language’ | 862 |
| 27.8 | Use Case: Creating Proxies, Adapters and Wrappers for Legacy C++ Applications | 864 |
| 27.8.1 | Alternative: SWIG (Simplified Wrapper and Interface Generator) | 871 |
| 27.9 | ‘Back to the Future’ Use Case: Calling C# Code from C++11 | 872 |
| 27.10 | Modelling Event-Driven Applications with Delegates | 876 |
| 27.10.1 | Next-Generation Strategy (Plug-in) Patterns | 877 |
| 27.10.2 | Events and Multicast Delegates | 881 |
| 27.11 | Use Case: Interfacing with Legacy Code | 886 |
| 27.11.1 | Legacy DLLs | 886 |
| 27.11.2 | Runtime Callable Wrapper (RCW) | 887 |
| 27.11.3 | COM Callable Wrapper (CCW) | 888 |
| 27.12 | Assemblies and Namespaces for C++/CLI | 889 |
| 27.12.1 | Assembly Types | 889 |
| 27.12.2 | Specifying Assembly Attributes in <code>AssemblyInfo.cs</code> | 890 |
| 27.12.3 | An Example: Dynamically Loading Algorithms from an Assembly | 891 |
| 27.13 | Summary and Conclusions | 895 |
| 27.14 | Exercises and Projects | 896 |

CHAPTER 28

| | | |
|--|--|------------|
| C++ Concurrency, Part I Threads | | 899 |
| 28.1 | Introduction and Objectives | 899 |
| 28.2 | Thread Fundamentals | 900 |
| 28.2.1 | A Small Digression into the World of OpenMP | 902 |
| 28.3 | Six Ways to Create a Thread | 903 |
| 28.3.1 | Detaching a Thread | 908 |
| 28.3.2 | Cooperative Tasking with Threads | 908 |
| 28.4 | Intermezzo: Parallelising the Binomial Method | 909 |
| 28.5 | Atomics | 916 |
| 28.5.1 | The C++ Memory Model | 918 |
| 28.5.2 | Atomic Flags | 920 |
| 28.5.3 | Simple Producer-Consumer Example | 922 |
| 28.6 | Smart Pointers and the Thread-Safe Pointer Interface | 924 |

| | | |
|-------|-----------------------------|-----|
| 28.7 | Thread Synchronisation | 926 |
| 28.8 | When Should we use Threads? | 929 |
| 28.9 | Summary and Conclusions | 929 |
| 28.10 | Exercises and Projects | 930 |

CHAPTER 29

| C++ Concurrency, Part II Tasks | | 935 |
|---------------------------------------|---|------------|
| 29.1 | Introduction and Objectives | 935 |
| 29.2 | Finding Concurrency: Motivation | 936 |
| 29.2.1 | Data and Task Parallelism | 936 |
| 29.3 | Tasks and Task Decomposition | 937 |
| 29.3.1 | Data Dependency Graph: First Example | 937 |
| 29.3.2 | Data Dependency Graph: Generalisations | 939 |
| 29.3.3 | Steps to Parallelisation | 940 |
| 29.4 | Futures and Promises | 941 |
| 29.4.1 | Examples of Futures and Promises in C++ | 942 |
| 29.4.2 | Mapping Dependency Graphs to C++ | 944 |
| 29.5 | Shared Futures | 945 |
| 29.6 | Waiting on Tasks to Complete | 948 |
| 29.7 | Continuations and Futures in Boost | 950 |
| 29.8 | Pure Functions | 952 |
| 29.9 | Tasks versus Threads | 953 |
| 29.10 | Parallel Design Patterns | 953 |
| 29.11 | Summary and Conclusions | 955 |
| 29.12 | Quizzes, Exercises and Projects | 955 |

CHAPTER 30

| Parallel Patterns Language (PPL) | | 961 |
|----------------------------------|--|-----|
| 30.1 | Introduction and Objectives | 961 |
| 30.2 | Parallel Algorithms | 962 |
| 30.2.1 | Parallel For | 963 |
| 30.2.2 | Parallel <code>for_each</code> | 964 |
| 30.2.3 | Parallel Invoke and Task Groups | 964 |
| 30.2.4 | Parallel Transform and Parallel Reduction | 967 |
| 30.3 | Partitioning Work | 967 |
| 30.3.1 | Parallel Sort | 970 |
| 30.4 | The Aggregation/Reduction Pattern in PPL | 971 |
| 30.4.1 | An Extended Example: Computing Prime Numbers | 973 |
| 30.4.2 | An Extended Example: Merging and Filtering Sets | 975 |
| 30.5 | Concurrent Containers | 977 |
| 30.6 | An Introduction to the Asynchronous Agents Library and Event-Based Systems | 978 |
| 30.6.1 | <i>Agents Library</i> Overview | 979 |
| 30.6.2 | Initial Examples and Essential Syntax | 980 |
| 30.6.3 | Simulating Stock Quotes Work Flow | 983 |
| 30.6.4 | Monte Carlo Option Pricing Using Agents | 985 |
| 30.6.5 | Conclusions and Epilogue | 985 |

| | | |
|--|---|-------------|
| 30.7 | A Design Plan to Implement a Framework Using Message Passing and Other Approaches | 986 |
| 30.8 | Summary and Conclusions | 989 |
| 30.9 | Exercises and Projects | 990 |
| CHAPTER 31 | | |
| | Monte Carlo Simulation, Part I | 993 |
| 31.1 | Introduction and Objectives | 993 |
| 31.1.1 | Software Product and Process Management | 994 |
| 31.1.2 | Who can Benefit from this Chapter? | 995 |
| 31.2 | The Boost Parameters Library for the Impatient | 995 |
| 31.2.1 | Other Ways to Initialise Data | 997 |
| 31.2.2 | Boost <i>Parameter</i> and Option Data | 999 |
| 31.3 | Monte Carlo Version 1: The Monolith Program ('Ball of Mud') | 1000 |
| 31.4 | Policy-Based Design: Dynamic Polymorphism | 1003 |
| 31.5 | Policy-Based Design Approach: CRTP and Static Polymorphism | 1011 |
| 31.6 | <i>Builders</i> and their Subcontractors (<i>Factory Method Pattern</i>) | 1013 |
| 31.7 | Practical Issue: Structuring the Project Directory and File Contents | 1014 |
| 31.8 | Summary and Conclusions | 1016 |
| 31.9 | Exercises and Projects | 1017 |
| CHAPTER 32 | | |
| | Monte Carlo Simulation, Part II | 1023 |
| 32.1 | Introduction and Objectives | 1023 |
| 32.2 | Parallel Processing and Monte Carlo Simulation | 1023 |
| 32.2.1 | Some Random Number Generators | 1025 |
| 32.2.2 | A Test Case | 1026 |
| 32.2.3 | C++ Threads | 1029 |
| 32.2.4 | C++ Futures | 1031 |
| 32.2.5 | PPL Parallel Tasks | 1031 |
| 32.2.6 | OpenMP Parallel Loops | 1032 |
| 32.2.7 | Boost Thread Group | 1032 |
| 32.3 | A Family of Predictor–Corrector Schemes | 1033 |
| 32.4 | An Example (CEV Model) | 1038 |
| 32.5 | Implementing the Monte Carlo Method Using the <i>Asynchronous Agents Library</i> | 1041 |
| 32.6 | Summary and Conclusions | 1047 |
| 32.6.1 | Appendix: C++ for Closed-Form Solution of CEV Option Prices | 1047 |
| 32.7 | Exercises and Projects | 1050 |
| Appendix 1: Multiple-Precision Arithmetic | | 1053 |
| Appendix 2: Computing Implied Volatility | | 1075 |
| References | | 1109 |
| Index | | 1117 |